Lesson Description - Strings

Let's learn about one of the core data types in Python: the `str` type.

Note:

\ - backslash

/ - forward slash

Documentation

- [strings (the `str` type)] https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str

Strings

Open a REPL to start exploring Python strings:

```
$ python3.7
```

We've already worked with a string when we created our "Hello, World!" program. We create strings using either single quotes (`'`), double quotes (`"`), or triple single or double quotes for a multi-line string.

```
>>> 'single quoted string'
'single quoted string'
>>> "double quoted string"
'double quoted string'
>>> '''
... this is a triple
... quoted string
... '''
'\nthis is a triple\nquoted string\n'
```

Strings also work with some arithmetic operators.

We can combine strings using the `+` operator and multiply a string by a number using the `*` operator:

```
>>> "pass" + "word"
'password'
>>> "Ha" * 4
'HaHaHaHa'
```

A string is a sequence of characters grouped together. We do need to cover the concept of an "Object" in object oriented programming before moving on. An "object" encapsulates two things 1) State & 2) behavior. For the built in types, the state makes sense because it's the entire contents of the object. The behavior aspect means that there are functions that we can call on the instances of the objects that we have. A function bound to an object is called a "method". Here are some example methods that we can call on strings:

`find` locates the first instance of a character (or string) in a string. This function returns the index of the character or string.

```
>>> "double".find('s')
-1
>>> "double".find('u')
2
>>> "double".find('bl')
3
```

`lower` converts all of the characters in a string to their lowercase versions (if they have one). This function returns a new string without changing the original, and this becomes important later.

```
>>> "TeStInG".lower() # "testing"
'testing'
>>> "another".lower()
'another'
>>> "PassWord123".lower()
'password123'
```

Lastly, if we need to use quotes or special characters in a string we can do that using the '\' character:

```
>>> print("Tab\tDelimited")
Tab     Delimited
>>> print("New\nLine")
New
```

```
Line
>>> print("Slash\\Character")
Slash\Character
>>> print("'Single' in Double")
'Single' in Double
>>> print('"Double" in Single')
"Double" in Single
>>> print("\"Double\" in Double")
"Double" in Double
```